

NO-A192 281

NEBULA SUPPORT SOFTWARE(U) ODYSSEY RESEARCH ASSOCIATES
INC ITHACA NY A SHENKER ET AL. OCT 87 RADC-TR-87-179
F30602-84-C-0182

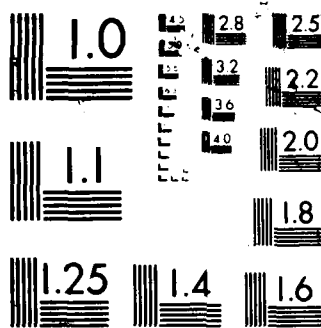
1/1

UNCLASSIFIED

F/G 12/5

NL





AD-A192 281

DTIC FILE COPY

RADC-TR-87-179

Final Technical Report

October 1987



NEBULA SUPPORT SOFTWARE

Odyssey Research Associates

Abraham Shenker, Leonard Silver and George Gesslein



APPROVED FOR PUBLIC RELEASE, DISTRIBUTION UNLIMITED

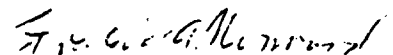
**ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700**

88 2 2 070

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-87-179 has been reviewed and is approved for publication.

APPROVED:



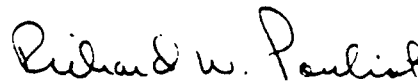
FREDERICK A. NORMAND
Project Engineer

APPROVED:



CHARLES E. ANDERSON, Lt Col, USAF
Acting Director of Command & Control

FOR THE COMMANDER:



RICHARD W. POULIOT
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COTC) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notice on a specific document requires that it be returned.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS N/A		
2a. SECURITY CLASSIFICATION AUTHORITY N/A			3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.		
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE N/A			5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-87-179		
4. PERFORMING ORGANIZATION REPORT NUMBER(S) N/A		6a. NAME OF PERFORMING ORGANIZATION Odyssey Research Associates, Inc.		6b. OFFICE SYMBOL (If applicable) COTC	
7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (COTC)		7b. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F30602-84-C-0182	
6c. ADDRESS (City, State, and ZIP Code) 1283 Trumansburg Road Ithaca NY 14850-1313		8a. NAME OF FUNDING / SPONSORING ORGANIZATION Rome Air Development Center		8b. OFFICE SYMBOL (If applicable) COTC	
8c. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700		10. SOURCE OF FUNDING NUMBERS		11. TITLE (Include Security Classification) NEBULA SUPPORT SOFTWARE	
12. PERSONAL AUTHOR(S) Abraham Shenker (Intermetrics), Leonard Silver, George Gesslein (ORA)		13a. TYPE OF REPORT Final		13b. TIME COVERED FROM Sep 84 to Jun 87	
14. DATE OF REPORT (Year, Month, Day) October 1987		15. PAGE COUNT 24		16. SUPPLEMENTARY NOTATION N/A	
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)			
FIELD		GROUP		SUB-GROUP	
12		05			
19. ABSTRACT (Continue on reverse if necessary and identify by block number) This is the final report for the Nebula Support Software contract, RADC contract F30602-84-C-0182. It provides a Unix-based capability on the VAX-11/780 which supported the development of software targeted for execution on the Nebula computer operating under the control of a native Unix system. This effort consisted of two tasks - the first was the retarget of the System V c-compiler and c-library to the Nebula Instruction Set Architecture (ISA), and the second was the port of the System V kernel to the Nebula Brassboard System developed by CDC. The code generated by the retargeted c-compiler will be run on the Nebula simulator and on the RADC Brassboard implementation of the Nebula ISA.					
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL Frederick A. Normand			22b. TELEPHONE (include Area Code) (315) 330-2925		22c. OFFICE SYMBOL RADC (COTC)

DD Form 1473, JUN 86

Previous editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED



A-1

1. Introduction

1.1 Identification

This report is the Final Report for the Nebula Support Software contract, RADC contract No. F30602-84-C-0182. It presents the results of the effort. The tasks and technical requirements for the Nebula Support Software effort are fully discussed in the Nebula Support Software Statement of Work (Revised January 11, 1984, PR No. B-4-3133).

1.2 Project Overview

The objective of the Nebula Support Software effort is to provide a Unix-based capability on the VAX-11/780 which supports the development of software targeted for execution on the Nebula computer operating under control of a native Unix system. This effort has been divided into two tasks:

1. Retarget the System V C compiler and C library for the Nebula ISA
2. Port the System V Kernel to the Nebula Brassboard System

The code generated by the retargeted C compiler will be run on the Nebula simulator and on the RADC Brassboard implementation of the Nebula Instruction Set Architecture.

2. Project Results

The retarget of the System V C compiler has been successfully completed. The C compiler is hosted on VAX/EUNICE, and generates assembly language compatible with Nebula assembler. The assembly language generated by the C compiler has been successfully assembled using the Nebula assembler, linked by the Nebula linker and executed under control of the Nebula simulator. Testing the compiler on the Brassboard implementation is complete.

2.1 Evaluation of Retarget Approach

The approach taken was to retarget the VAX System V Release 2.0 C compiler. This compiler was selected because it is easily retargeted and totally compatible with the System V Kernel and utilities that will be rehosted. The selection of the System V compiler constitutes an exception to the SOW requirement (SOW paragraph 4.1.1), that the compiler be the same as the VMS/Unix development environment. The System V C compiler and EUNICE C compiler have both descended from the Johnson Portable C compiler and the languages and semantics of the two compilers are highly compatible.

The VAX System V C compiler is a version of the Johnson Portable C Compiler. This compiler is entirely written in C and designed to be easily retargeted. The compiler is organized along the lines of machine dependent and machine independent modules. Most of the instruction generation dependencies are encapsulated into a set of code generation tables.

We found the modification of the Johnson Portable C Compiler to be straight forward. There are aspects of the Nebula Architecture that are sufficiently different from most conventional architectures that exposed some erroneous assumptions of the machine independent portions of the compiler. A few of the allegedly machine independent modules had to be modified slightly for the Nebula retarget. A large portion of the effort was rewriting the code generation tables. This task was greatly simplified by the similarity of the VAX and Nebula instruction sets and argument addressing modes. The register allocator modifications also comprised a major component of the retarget effort. The register allocator was particularly machine dependent because the VAX C compiler had implicit assumptions regarding the run-time model and properties of general purpose registers.

The C library was extremely easy to retarget with the exception of the setjmp/longjmp library routines. These library routines had a strong dependence on the VAX run-time model and the properties of the run-time stack. These routines had to be redesigned and recoded from scratch. Most of the other C library routines only required recompilation.

The development environment selected to perform the retarget was also a strong factor in the success of this part of the effort. Sun workstations (Motorola 68000 based Unix 4.2bsd) provided an ideal program development environment. The robust set of debugging, configuration management and communication tools available on the workstations greatly reduced the effort involved in modifying, debugging and testing the retargeted compiler. The compiler was entirely debugged using the workstation and then rehosted to the VMS/EUNICE environment for testing. The C compiler generated code was tested by using the Nebula Assembler and Nebula Simulator. The RADC supplied Nebula support tools

performed very well, and presented only few small obstacles to overcome.

2.2 Evaluation of Nebula ISA for C

The Nebula ISA instruction set provides a good support for the C language. The rich set of operand addressing modes supports C data types nicely. The generalized operand size rules greatly improves the code generated for expressions involving different data representations. The register 1 stack provides good support for local variables. In general, the code quality (size and instruction count) for C expressions equaled or exceeded that of the VAX targeted compiler.

The Nebula procedure model does not optimally suit the C language function model. C function arguments must be passed by value. Since the parameters must be copied, there is little advantage to using the Nebula parameter passing mechanism. Although the Nebula does not provide full hardware support for C function linkage, the resulting call sequence is only slightly more cumbersome than that on the VAX, and the implemented call sequence is not expected to cause serious performance problems.

The following weaknesses of the ISA were found when designing the run-time model for the C programming language:

1. Support for subroutine linkage (jsr and rsr) is very weak. There is no efficient way to save the entire register set of registers into memory or on the register 1 stack. This severely limits the utility of subroutines as an alternative procedure linkage model design.
2. The push unsigned instruction is conspicuously missing from the ISA.
3. An indirect register index operand addressing mode would be very useful. The same effect can be generated using either an intermediate register or a compound addressing mode. An indirect register index addressing mode would be very useful for languages like C that rely heavily upon pointers.

2.3 Benchmarking Results

The benchmark tests demonstrated that the retargeted compiler produces code equal to the Vax C compiler.

2.4 Summary of Retarget Effort

The VAX System V C compiler and C library have been successfully retargeted to the Nebula. The Nebula ISA provides a very good support for C expressions and data access. The code quality

produced by the Nebula C compiler is very close to the quality of that produced by the VAX compiler.

2.5 Summary of Unix Port

The Unix system was ported to the Nebula Brassboard. All non-I/O functions have been successfully tested. All I/O functions were tested. Many of the tests failed due to hardware problems. Nebula Unix successfully executes non-I/O dependent applications.

3. Unix Port Technical Approach

The following sections discuss the general technical approach to be followed in performing each of the tasks outlined above.

3.1 Unix Kernel Port

Porting the Unix Kernel to the Nebula will involve rewriting the machine dependent parts of the VAX Unix Kernel. Some of the machine dependent code is VAX assembler source code. The VAX assembler source code will be recoded into Nebula assembler source code. The great majority of the machine dependent code is written in C, and will have to be modified by hand. There are a few parts of the kernel which are largely machine independent. These routines will be recompiled and linked with the machine dependent routines to produce an executable Unix kernel. A summary of the kernel modules being rehosted and the work that is anticipated for porting each to the Nebula is provided as Appendix A.

3.1.1 Design and Implementation Goals

In designing the modifications to the Unix Kernel, several goals are being pursued. These goals have, and will serve as guidelines to aid in evaluating critical design decisions and selecting implementations.

Implementation Dependencies

It is a goal to produce a Nebula/Unix system that can be easily ported to another implementation of the Nebula (1862B) architecture. The Nebula Architecture definition does not specify the entire functionality of a Nebula computer. Many features are defined to be Implementation Dependent. Operating systems must interact with many of these implementation dependencies, thus making it impossible to write a totally implementation independent Nebula/Unix Kernel. The modifications will be performed so as to minimize the work needed to port the Brassboard Nebula Unix to another hardware implementation. This will be done by clearly commenting all implementation dependencies in the source code. An attempt will be made to organize implementation dependencies so that they may be easily identified and modified.

Use of Nebula Hardware

It is a goal to use the Nebula hardware functionality whenever appropriate. Although it is a goal to make the Nebula/Unix kernel implementation independent, design choices will exploit the hardware implementation of the Brassboard system. The

exploitation will result in better performance and will not be done at the expense of the implementation independence goal. This approach will aid in the evaluation of the Nebula hardware and the Brassboard implementation.

Kernel Functionality

It is a goal to maintain compatibility between the VAX System V and the Nebula Unix kernels. Compatibility will reduce the effort required to rehost other System V system commands and utilities to the Nebula.

Coding Conventions

It is a goal to adhere to the same coding conventions and practices followed in the original System V Unix. These conventions include global code organization, file, variable, function naming conventions and code formatting conventions. These standards will aid programmers familiar with VAX System V Unix internals to maintain the Nebula Unix Kernel.

Future System Enhancements

It is a goal to implement the Nebula Unix Kernel so as to facilitate the addition of functionality; particularly Real-Time and Secure Operating system features. Capacity limitations will be selected so as to allow for system expansions and modifications.

3.1.2 Architectural Differences

The work that will be performed to modify the Unix Kernel can be categorized along the architectural differences between the Nebula and the VAX machines. The two machines are dissimilar in their implementation of virtual memory, I/O, procedure model and exception handling. Their non-privileged instruction sets, addressing modes and interrupt handling structure are very similar.

Some of the modifications, such as the invocation of privileged or Nebula ISA specific instructions and functions will certainly be coded in assembler language. The great majority of the modifications will be performed by modifying the C code.

Byte Ordering

The VAX and the Nebula are both byte addressable machines. The VAX addresses a word by the address of the least significant byte. The Nebula addresses a word by the address of the most significant byte. It is possible to write C code that is dependent upon the addressing of bytes within a word. There is C code in the Unix Kernel that assumes the byte ordering of the VAX.

This code will be modified to reflect the Nebula byte ordering.

Procedure Model

The difference between the Nebula and VAX procedure model centers around the Nebula procedure context stack. The VAX Unix Kernel occasionally manipulates the VAX User stack to modify a process's flow of control. The fact that the general registers are allocated on the Nebula context stack also creates a rehosting problem. The implementation of the Unix process model has been redesigned to take the implications of the Nebula context stack into account. These modifications will effect context switching, virtual memory management, signal handling, non-local gotos (longjmp) and process management functions.

Manipulation of the Nebula context stack is extremely implementation dependent. The Nebula architecture description does not fully define the format of the context stack content. Similarly the context stack caching mechanisms are implementation dependent. This implies that manipulating data in Nebula context stack would be extremely implementation dependent. The caching properties could pose very serious performance problems keeping context memory data consistent with context cache. The Unix Kernel will be modified so as to avoid manipulating the contents of the context stack whenever possible.

Memory Management

The differences between the VAX and Nebula memory management hardware are responsible for the most prevalent modifications needed in performing a rehost from the VAX to the Nebula. The memory mapping mechanism, protection mechanisms and virtual address interpretation are very different. Manipulation of the memory management hardware is an integral part of Unix Kernel initialization, process creation, process switching, memory allocation, memory trap handling, I/O operations and process tracing implementations. Since the impact of the memory management hardware is so broad, this area is expected to comprise the largest portion of the rehosting effort.

Input/Output

The I/O systems of the Nebula and VAX are very different. The Unix developers have been very careful to isolate the parts of the kernel that perform machine dependent I/O operations. The interface and structure of the code that performs machine dependent I/O operations is flexible and very well defined. The effort will consist of development of the device dependent drivers and modification of the interface routines to handle the Nebula/Unix process memory management. Modification and implementation of these modules will be straight-forward.

SVC, Traps and Exceptions

The Supervisor Calls (SVC), Traps and Exceptions of the Nebula are different from the VAX. The assembler code that handles these events funnels the handling of the events through a single C procedure called trap. This minimizes the machine dependence of these functions. It also provides a simple and uniform state when entering the Kernel. Rehosting to the Nebula will require writing of the system call and entry routines in assembler language and modification of the trap function.

Timer Support

The timer support of the Nebula is very close to that of the VAX. These modification will mostly entail changing code that references VAX dependent hardware registers to Nebula dependent hardware registers.

3.1.3 Kernel Preparation and IPL

The Nebula System V Kernel image will be created using similar technique as on the VAX. The Kernel will be compiled, assembled and linked on the VAX. The Nebula C compiler developed as the first part of the project will be used to compile all C code. The Nebula Assembler supplied by RADC will be used to assemble all output from the C compiler as well as the assembler language modules hand written as part of the Unix porting effort. Each object module produced by the Nebula assembler will be converted into the System V common object module (a.out) format. This conversion will be performed by a EUNICE hosted tool developed as part of this effort. These modules will be linked by the System V linker (ld) to produce the Kernel executable image. The tools used in this process will be the same tools used to generate programs for execution under Nebula/Unix Kernel control.

On the VAX, the computer loads a small program into VAX memory called the Bootstrap Loader. The VAX/Unix Bootstrap Loader then locates and loads the Unix Kernel from the a.out image on the root file system and passes control to the Kernel. On the Nebula, the Nebula IPL feature will be used to load a Bootstrap Loader. The Nebula/Unix Bootstrap will locate and load the Unix Kernel using the same technique as on the VAX.

3.1.4 Kernel Debugging

The Unix Kernel is a large multi-task program with many subtle real-time interactions. The most difficult parts to debug will be the low-level machine dependent initialization routines. These routines will be partially debugged using the Nebula simulator. This will greatly reduce the initial effort of getting the Nebula/Unix Kernel operating on the Brassboard system.

Unix has many self-consistency checks throughout the Kernel. If one of these checks fails, a Panic usually occurs. A Panic prints relevant information about the error on the console and dumps main storage. A dump can be examined with Unix tools (crash and adb) to determine the exact state of the operating system when the panic occurred. These tools will be modified to help debug the Nebula Unix kernel. They will be hosted on the VAX/EUNICE system.

3.2 Nebula I/O Channel Drivers

Because the Nebula I/O structure is significantly different from that of the VAX, the I/O portion of the Unix rehost will require modifications to the code that handles I/O on the VAX. The conversion of I/O routines for a generalized disk controller, a terminal, a console, and memory devices will be undertaken. The approach to making these modifications will be to replicate the functionality provided on the VAX on the Nebula Brassboard.

The console on the VAX makes use of specific registers and addresses designed for the console; there is a separate controller for the console. The Nebula has no such facility and the console will have to be connected via the Serial Point to Point (SPP) Channel on the Nebula. To do this will require modifying the cons.c and ttl.c modules so that they will work on the SPP. The SPP is significantly different from the VAX Massbus in that it is a programmable controller which will support much of the processing to be done in the controller rather than in the CPU as is the case on the VAX.

The terminals for the Nebula will also be connected via the SPP. On the VAX, terminals are usually attached via a Unibus device. The Unibus device is controlled by a device driver along with the ttl.c and tty.c terminal handling modules. On the Nebula the terminals will be connected directly to the machine through the SPP. This will require modifying the drivers and terminal handling modules so that they will control the SPP channel. As with the console, much of the I/O processing can be done by the programmable controller.

The memory devices, which perform I/O on areas of memory, will have to be modified to make use of the Nebula memory mapping. On the VAX, virtual addresses are translated by the I/O routines to their physical location. A similar approach will be used for the Nebula in that translation will be done by the I/O routines using knowledge of the Nebula memory mapping scheme.

A disk system will be provided which will simulate a block disk device using the Nebula memory space reserved for this purpose upon Unix Kernel initialization. Alternatively a disk can be emulated in a remote computer (like the VAX) using a remote disk connected via the SPP. There will be provisions for a

directly connected disk interfaced via the PPP channel. It may not be possible to test PPP channel implementation if the hardware is not available.

3.3 Cross Development Toolset

The Cross development toolset will be provided that will allow compilation of C programs on the VAX/EUNICE system and execution on the Nebula Brassboard system. Programs will be automatically prepared on the VAX/EUNICE development system by invoking a command procedure (shell script) to compile, assemble, convert, link and install the C program.

The C compiler has been developed as part of this effort. The Nebula Assembler language generated by the C compiler is completely compatible with the standard Nebula Assembler provided by RADC. The Nebula Object modules produced by the Nebula Assembler will be converted by the Nebula Object module converter. This converter is currently being developed as part of this effort, and will be hosted on VMS/EUNICE. It will produce System V Common Object Module Format object modules. These modules will be linked by the rehosted System V linker (ld) to form a System V executable image. This image will also be in Common Object Module Format, but will have the external references resolved.

The executable image will be installed in the Nebula/Unix file system. This can be performed in one of two ways, depending upon the system configuration. The Nebula/Unix file system will reside either in Nebula main memory storage, or on a magnetic disk storage. If the file system is to reside in Nebula main memory, an entire Unix file system will be built on the EUNICE system and down-loaded as part of the Nebula/Unix IPL procedure. If the file system is to reside on a magnetic disk storage device, the file system will be built on the EUNICE system and installed to serve as the initial root file system. Since there is currently no directly attached disk storage for the Brassboard system, another computer will emulate a disk storage device over a dedicated Serial Point to Point communications link. The computer emulating the disk drive will maintain an image of the file system in a disk file. I/O requests from the Nebula will be translated into I/O requests to it's local file system.

It will also be possible to down-load a program over a Serial Point to Point communications line, while the Unix system is running. This will be accomplished by a procedure similar the that used to down-load to the CPP, except the receiving computer will be the Nebula.

3.4 Bootstrap Toolset

A set of tools will be needed to generate and examine a boot-able root file system. The Unix utilities including mkfs, fsdb, fsck will be hosted on the VMS/EUNICE system and used to generate the image of a boot-able file system in a VMS/EUNICE file. These utilities allow the construction, interactive editing and consistency checking of the Unix file system. Depending upon the availability of a disk storage device, the image will either be down-loaded into a Nebula disk storage device, or incorporated as part of the Nebula/Unix Kernel image by the Bootstrap program.

3.5 Nebula Hosted Toolset

The Nebula hosted toolset will be derived from the standard System V toolset. Since these utilities are largely machine independent and written in C, they may be rehosted to the Nebula by recompilation of the C source code and installation into the Nebula file system. The toolset will include only those tools which are either machine independent, or are vital to the normal operation of Nebula/Unix. A summary of the non-kernel Unix tools to be ported to Nebula is presented in Appendix B. It should be noted that the ability to access the entire toolset at once may be impossible due to memory or disk limitations.

4 Functionality of the Ported Unix

The Nebula System V Kernel will provide a subset of the VAX System V Kernel functionality. Many of the modules not ported from the original VAX System V subset have been deleted because they support devices only available on the VAX. The subset has been selected so as to support all the functionality specified in of the statement of work (paragraph 4.2.1).

4.1 Kernel Subset

The statement of work specifies that the contractor should define a subset of the Unix Kernel to port to the Nebula hardware (paragraph 4.2). Some of the functionality of the VAX Unix Kernel will not be ported to the Nebula because the hardware support for is not available on the Nebula. There are a few large optional subsystems of the System V Kernel that are not required by the statement of work. The following features of the VAX System V Kernel will not be fully supported by the ported Nebula/Unix Kernel:

1. The ptrace(2) system call shall be limited. Certain accesses of the traced child's memory space (Context stack) will not be allowed. Full access of this system call on the Nebula would jeopardize system security and system integrity.
2. Nebula/Unix will not support the X25 network communications protocol.
3. The old file system format (512 byte block size) will be implemented, but not tested. The new file system format (1024 byte block) yields larger capacity and better performance. The old file system is available on the VAX strictly for backward compatibility.
4. The optional sharedmemory system calls will be implemented, but not fully tested.
5. The optional semaphore system calls will be implemented, but not fully tested.
6. The optional message system calls will be implemented, but not fully tested.
7. The optional virtual terminal interfaces will be implemented, but not fully tested.

If unimplemented features are invoked by a user program, error status codes will be returned.

4.2 Unix Extensions

In addition to the standard VAX System V Unix Kernel functionality, the Nebula/Unix will provide a capability for:

1. Memory resident file system.
2. Remote magnetic disk emulation.

These features will be added to the Kernel to facilitate testing and utility of the Nebula/Unix port in lieu of access to a directly connected high speed disk device.

The memory resident file system will be a region of physical memory reserved for virtual disk device. The virtual disk device will be a block-type special device suitable for mounting a file system. This device will be used to test file system functionality in the absence of a directly connected magnetic disk storage device.

The remote disk device will be a block-type special device that will service I/O requests by communicating with a remote machine emulating a disk drive. The communications will take place over a Serial communications line. Note that this approach will greatly augment the utility of the Nebula/Unix system until a directly connected magnetic disk storage device becomes available. The relatively low speed of the Serial interface will have a serious negative effect upon the Unix system performance.

Nebula/Unix Hosted Toolset

Appendix A summarizes Unix tools that were ported to the Nebula/Unix system. The following tools are executable on the Nebula/Unix system except for I/O problems.

at	-- execute commands at a later time
awk	-- pattern scanning and processing language
banner	-- make posters
basename	-- deliver portions of path names
bc	-- arbitrary-precision arithmetic language
cal	-- print a calendar
cat	-- concatenate and print files
cd	-- change directories
chmod	-- change mode
chown	-- change owner
chroot	-- change root directory for a command
cmp	-- compare two files
comm	-- select or reject lines common to two sorted files
cp	-- copy files
cron	-- clock daemon
crypt	-- encode/decode
date	-- print and set the date
dc	-- desk calculator
dd	-- convert and copy a file
df	-- report number of free disk blocks
diff	-- differential file comparator
diff3	-- 3-way differential file comparison
dircmp	-- directory comparison
du	-- summarize disk usage
echo	-- echo arguments
ed	-- text editor
env	-- set environment for command execution
expr	-- evaluate arguments as an expression
factor	-- factor a number
file	-- determine a file type
find	-- find files
getopt	-- parse command options
getty	-- set terminal type, modes, speed, and line discipline
grep	-- search a file for a pattern
id	-- print user and group IDs and names
init	-- process control initialization
join	-- relational database operator
kill	-- terminate a process
killall	-- kill all active processes
line	-- read one line
ln	-- link files
login	-- sign on
logname	-- get login name
ls	-- list contents of a directory
machid	-- provide truth value about your processor type
msg	-- permit or deny messages

```
mkdir      -- make a directory
mount      -- mount and dismount file system
mv         -- move files
ncheck     -- generate names from i-numbers
nice       -- run a command at low priority
nl         -- line numbering filter
nm         -- print name list of common object file
nohup      -- run a command immune to hangups and quits
od         -- octal dump
pack       -- compress and expand files
passwd     -- change login password
paste      -- merge same lines of several files or subsequent
              lines of one file
pg         -- file perusal filter for soft-copy terminals
pr         -- print files to standard output
ps         -- report process status
pwd        -- working directory name
rm         -- remove files
rmdir      -- remove directories
sed        -- stream editor
setmnt     -- establish mount table
sh         -- shell, the standard command programming language
shutdown   -- terminal all processing
size       -- print section sizes of common object files
sleep      -- suspend execution for interval
sort       -- sort and/or merge files
stty       -- set the options for a terminal
su         -- become super-user or another user
sum        -- print checksum and block count of a file
sync       -- update the super block
tail       -- deliver the last part of a file
tee        -- pipe fitting
test       -- condition evaluation command
touch      -- update access and modification times of a file
tput       -- query terminfo database
tr         -- translate characters
true       -- provide truth values
tty        -- get the name of the terminal
umask      -- set file-creation mode mask
uname      -- print name of current Unix system
uniq       -- report repeated lines in a file
units      -- conversion program
wait       -- await completion of process
wall       -- write to all users
wc         -- word count
who        -- who is on the system
write      -- write to another user
xargs      -- construct argument lists and execute command
```

A decorative border with a repeating geometric pattern surrounds the entire page. A smaller, similar border frames the central text area.

MISSION of Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control, Communications and Intelligence (C³I) activities. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of C³I systems. The areas of technical competence include communications, command and control, battle management, information processing, surveillance sensors, intelligence data collection and handling, solid state sciences, electromagnetics, and propagation, and electronic, maintainability, and compatibility.

END

DATE

FILMED

6-1988

DTIC